Software Design Document

for

Version 1.0

Prepared by

Group Name:

Lilea Anamaria Adriana   CEN 4.S1B

Stan Ana-Maria Bianca   CEN 4.S2B

TABLE OF CONTENTS

1 Introduction

1.1 Purpose

The purpose of the Software Design Document is to provide a description of the design of a system fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to be built. The Software Design Document provides information necessary to provide description of the details for the software and system to be built.

1.2 Scope

The purpose of this software is a web application designed to manage a clinic for animals. The user interface is different according to either doctors or pet owners/clients.
The pet owner/client can login into the application, to add an appointment, to delete it and also can add a feedback.
The doctor can login into the application, to add an appointment for client, to add clinic services like vaccination etc, price for it;
The application contains some general pages for all users like contact, about the clinic and of course a welcome page.

► The reason for starting this website is to help all pets in our area live happy, healthy lives. All of us at Healthy vet Clinic love animals and are trained and ready to take care of any problem that comes along. We believe that the focus of veterinary service and medicine is the bond between our clients and their pets. Our objectives of this business are:

- To provide care and protection from disease
- Help all of our patients live a long and healthy life
- To provide any advice that a client may have about their pet
- To provide an affordable full time medical service to all who are in need
- To provide more than medical help for pets such as boarding, tags, toys, and supplies

► With my application help, the clinic, its doctors and its staff embrace apps for everything from marketing to communicating with clients and informing them about the dangers of diseases to looking up patient records and performing basic office tasks.

- ► The Vet App also enables clients to request appointments, check on contact(phone number, email etc), get emergency instructions and find links to a clinic’s website and to another.

- ► The system has several modules. Major modules of the app are as follows:
- ► **Home page**: Welcome for anyone who access the site(doctors/users)
- ► **Service:** Add, Modify or Delete service information- this page is seen by doctors
- ► **Appointment** : This page is seen by doctors and the users that sign into our app.The pet owner can make an appointment and if he wants to delete it, he needs to confirm before deletes it.
- ► **About** : Information and pictures about our clinic and the user can add a comment only if they are logged in and can be seen by doctors and another users logged in.

1.3 Overview

The Software Design Document is divided into 8 sections with various subsections. The sections of the Software Design Document are:

1 INTRODUCTION

2 SYSTEM OVERVIEW

3 SYSTEM ARCHITECTURE

4 DATA DESIGN

5 COMPONENT DESIGN

6. HUMAN INTERFACE DESIGN

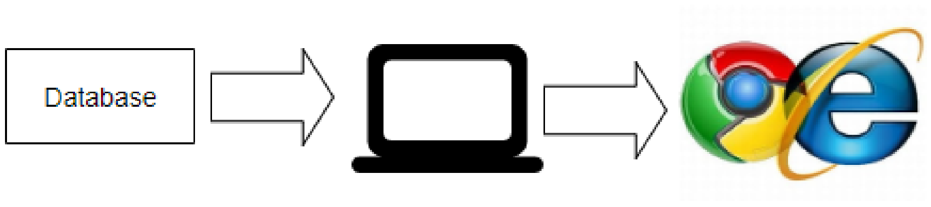
7 REQUIREMENTS MATRIX

8. APPENDICES

2.SYSTEM OVERVIEW

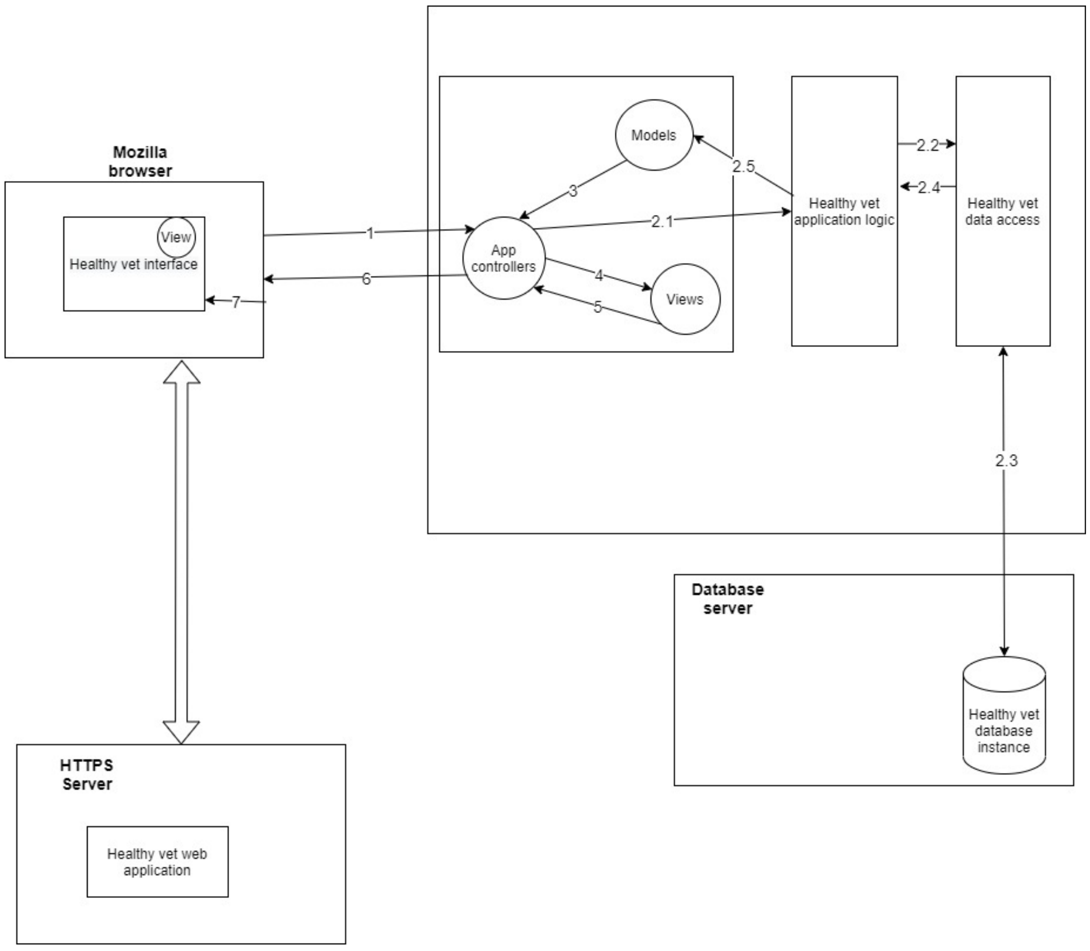
The Healthy Vet system is a web application designed to manage a clinic for animals. This system is intended for a veterinary clinic that can provide services for pets. With the help of this application the petowners can request appointments, check on contact(phone number, email etc), get emergency instructions and find links to a clinic’s website and to another. And at the same time the doctors and its staff embrace apps for everything from marketing to communicating with clients and informing them about the dangers of diseases to looking up patient records and performing basic office tasks.

Below is a diagram of the Healthy Vet system which illustrates the interactions between the server and the clients.



3.SYSTEM ARCHITECTURE

3.1 Architectural Design

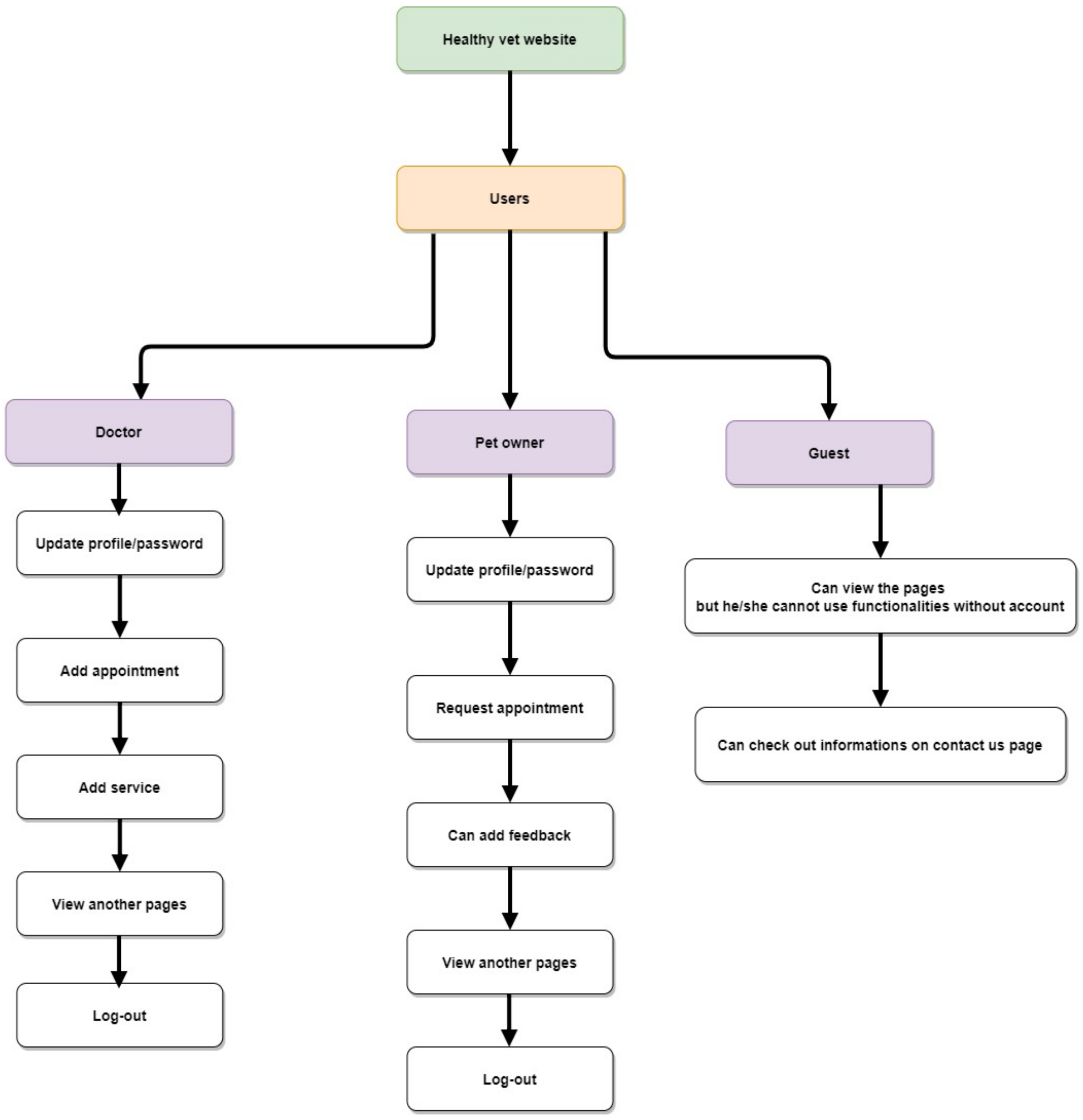


The system architecture that is used for the application **is MVC pattern (model-view-controller).**This architectural system considers 3 roles: model,view and controller.
The **model** is an object that represents some information about the domain. It's a nonvisual object containing all the data and behavior other than used for the UI.
The application will have 4 models:

- Appointment model - is used to add an appointment and to update it.
- Pet owner model - with this model, the client can add a feedback.
- Doctor model - is used for doctor to add one or more services.
- Create role model - it is used to add 2 roles for the 2 users: doctor and pet owner for creating an account. The account will be stored in the database which is created with entity framework.

The **view** represents the display of the model in the UI. Thus, if our model is a customer object our view might be a frame full of UI widget or an HTML page rendered with information from the model.
The view is only about display of information; any changes to the information are handled by the third member of the MVC trinity: the controller.
The **controller** takes user input, manipulates the model, and causes the view to update appropriately. In this way UI is a combination of the view and the controller.
The app will have 3 controllers: Appointment, Doctor and Pet owner controller.
For the website database it will be used **Entity Framework. EF** is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations.

3.2 Decomposition Description



3.3 Design Rationale()

The approach for this website architecture has been selected because it is probably the most common; it is usually built around the database.

The architecture is arranged so the data enters the top layer and works its way down each layer until it reaches the bottom, which is usually a database. Along the way, each layer has a specific task, like checking the data for consistency or reformatting the values to keep them consistent. It is the standard software development approach offered by most of the popular web frameworks and it is a layered architecture. Just above the database is the model layer, which often contains business logic and information about the types of data in the database. At the top is the view layer, which is often CSS, JavaScript, and HTML with dynamic embedded code. In the middle, you have the controller, which has various rules and methods for transforming the data moving between the view and the model.

The advantage of a layered architecture is the separation of concerns, which means that each layer can focus solely on its role. This makes it:

- Maintainable
- Testable
- Easy to assign separate "roles"
- Easy to update and enhance layers separately

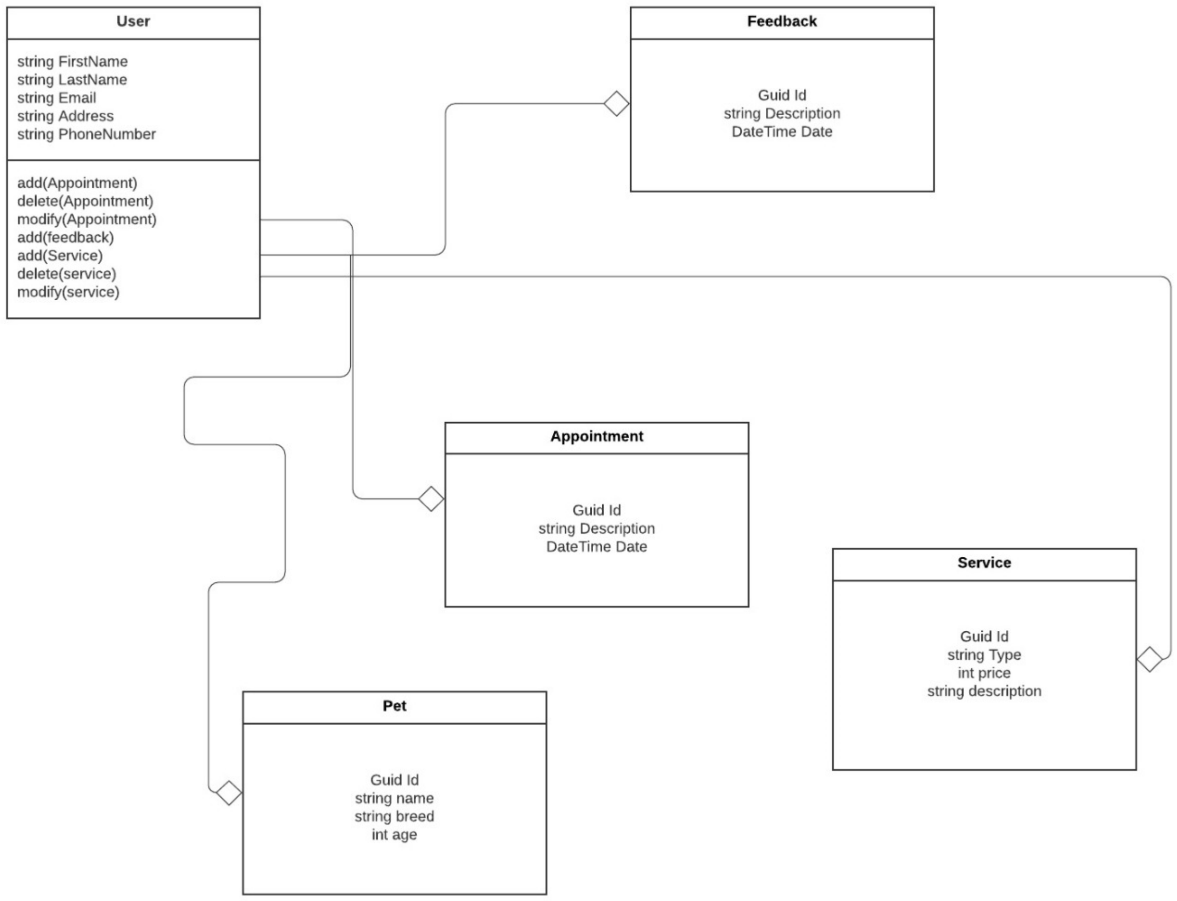
Proper layered architectures will have isolated layers that aren't affected by certain changes in other layers, allowing for easier refactoring. This architecture can also contain additional open layers, like a service layer, that can be used to access shared services only in the business layer but also get bypassed for speed.

Best for:

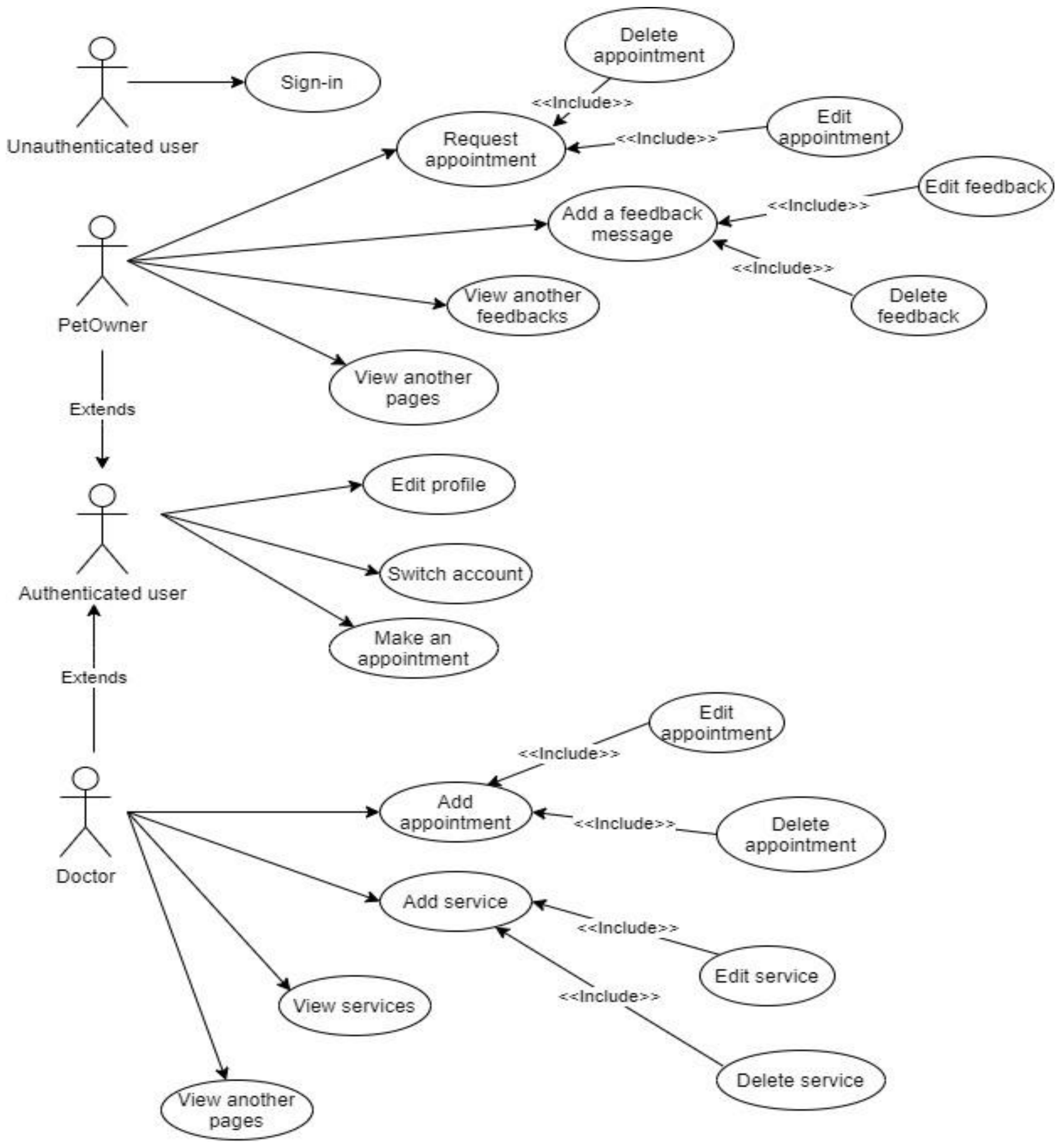
- New applications that need to be built quickly
- Enterprise or business applications that need to mirror traditional IT departments and processes
- Teams with inexperienced developers who don't understand other architectures yet
- Applications requiring strict maintainability and testability standards

5. Component Design

- **Class diagram**

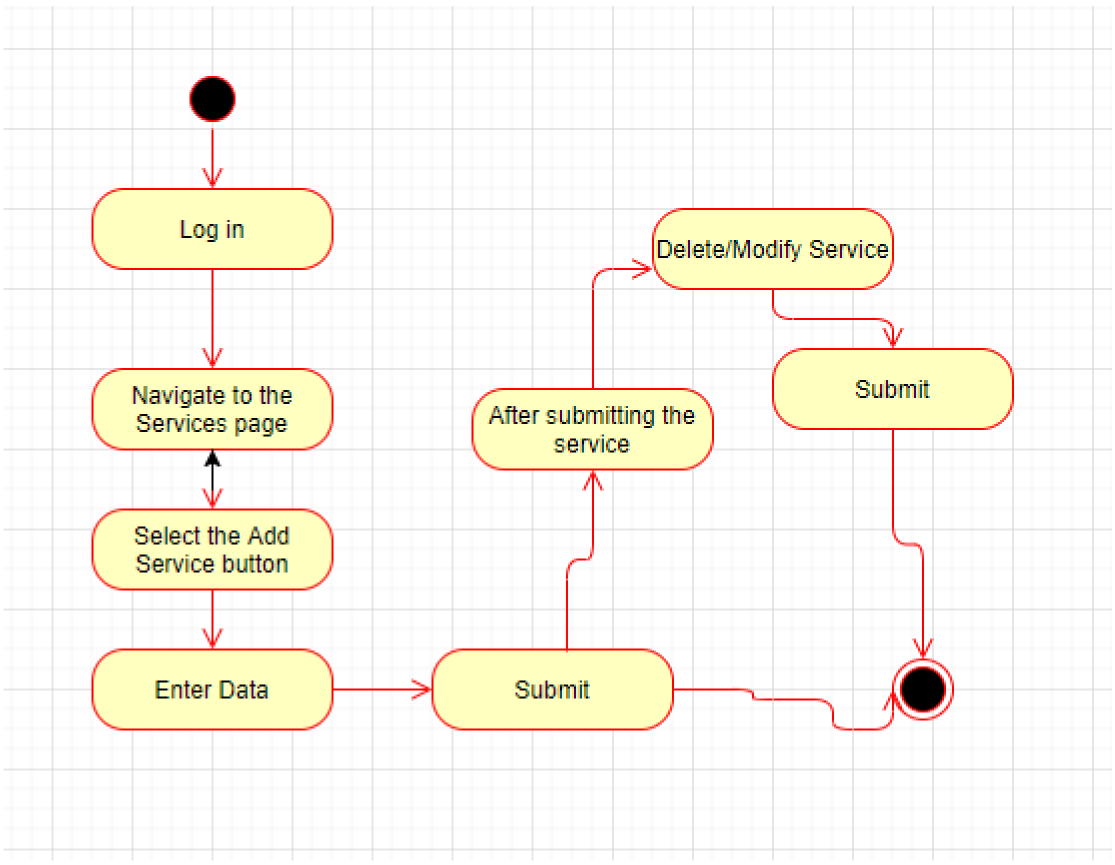


- **Use case-diagram**

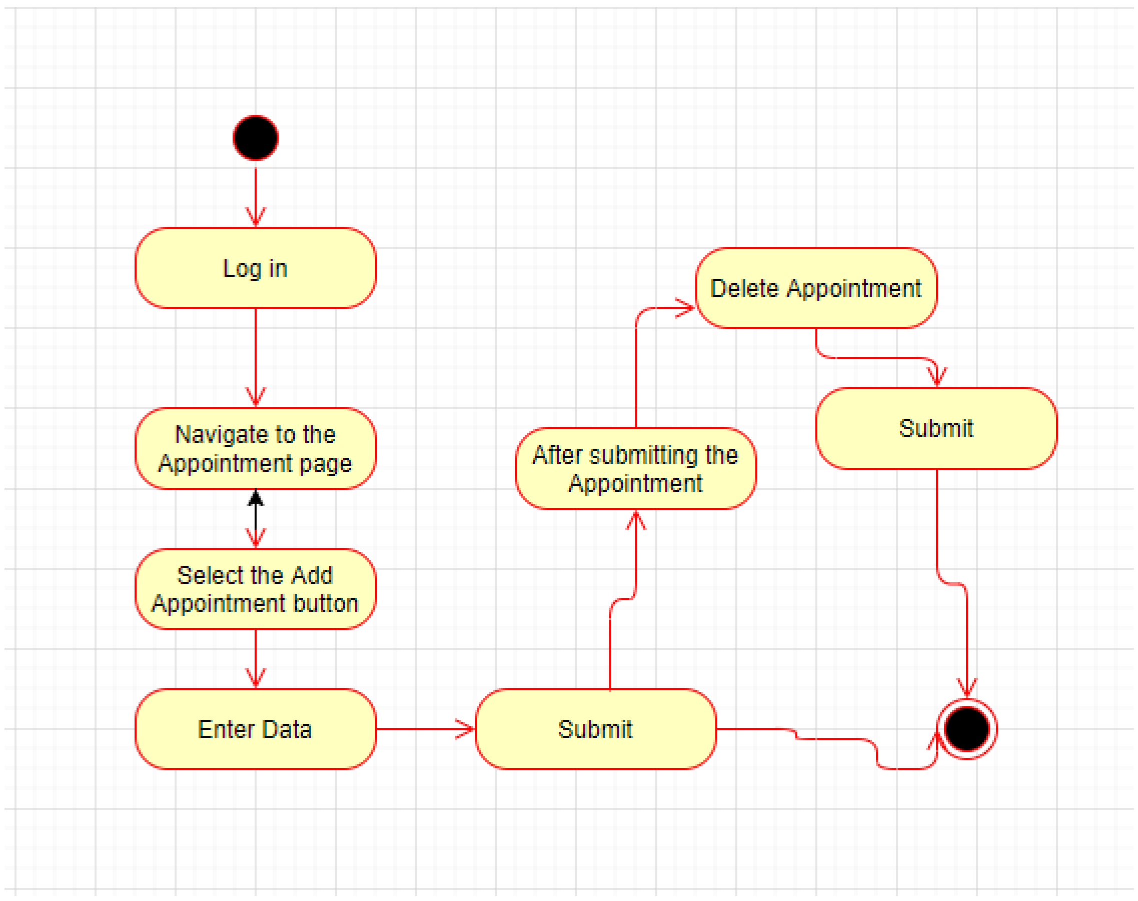


Activity diagrams

- **Add a service**

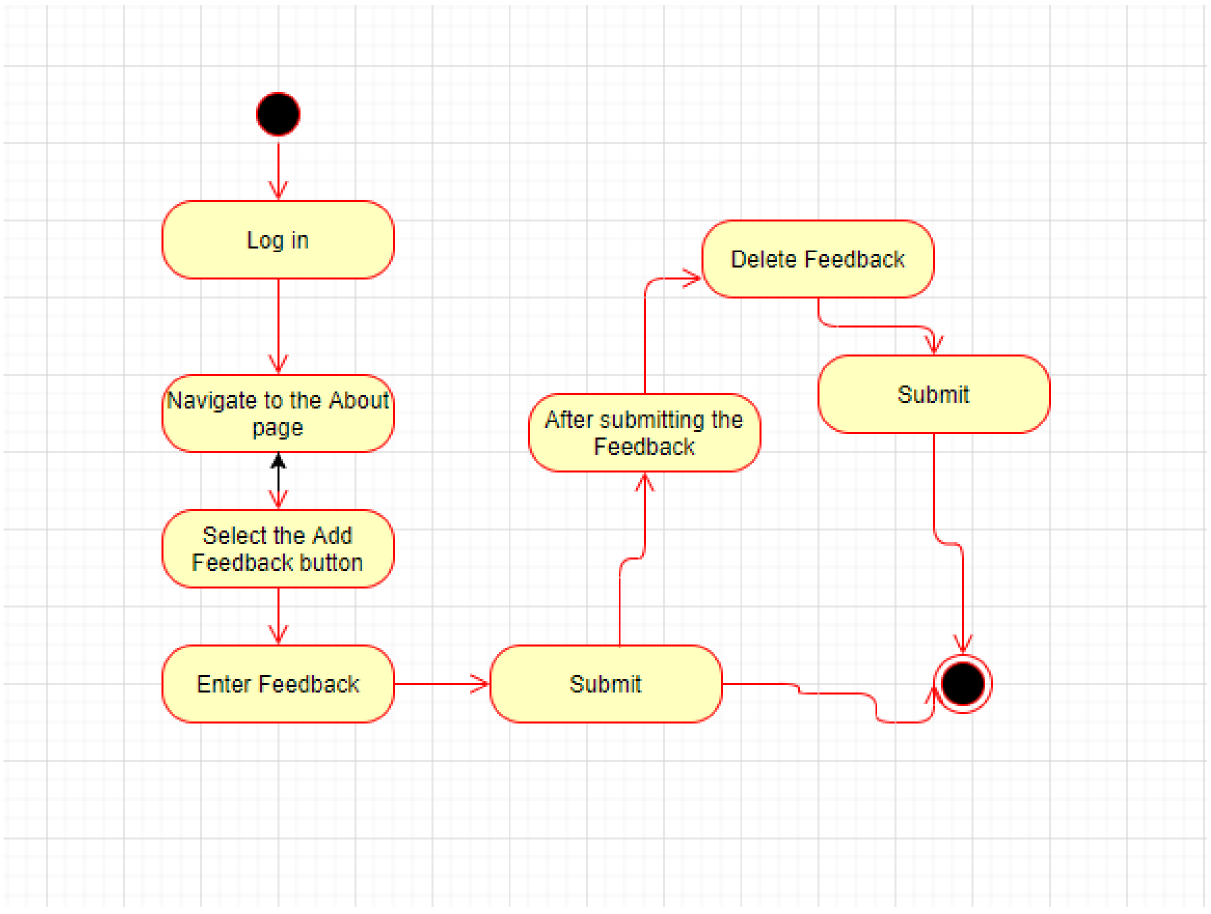

Log in
Navigate to the Services page
Select the Add Service button
Enter Data
Submit
After submitting the service
Delete/Modify Service
Submit

- **Add an appointment**

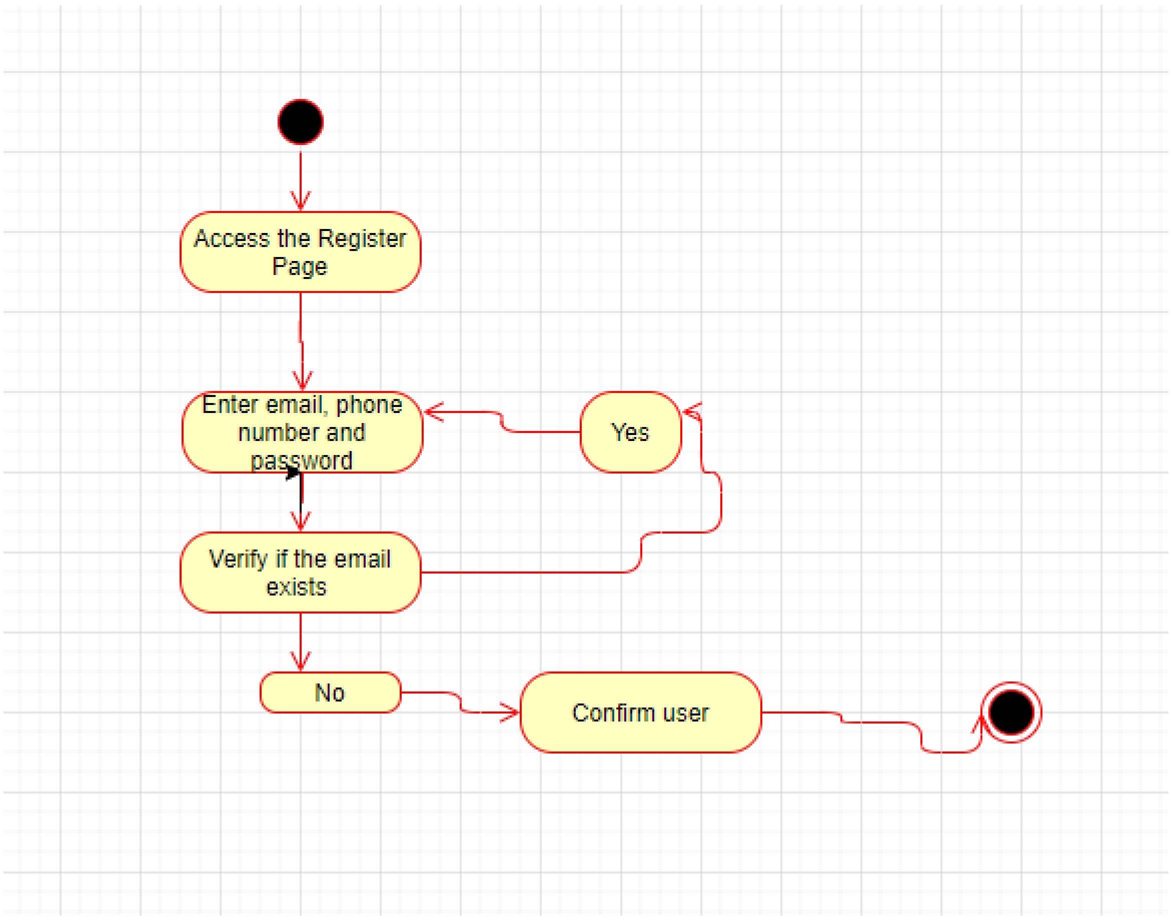


Log in
Navigate to the Appointment page
Select the Add Appointment button
Enter Data
Submit
After submitting the Appointment
Delete Appointment
Submit

- **Add feedback**


Log in
Navigate to the About page
Select the Add Feedback button
Enter Feedback
Submit
After submitting the Feedback
Delete Feedback
Submit

- **Register a user**


Access the Register Page
Enter email, phone number and password
Yes
Verify if the email exists
No
Confirm user

8. APPENDICES

This section is optional.